Science of Computer Programming 89 (2014) 2–22

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

A verified algebra for read–write Linked Data

Ross Horne [a,b,*], Vladimiro Sassone [a]

[a] Electronics and Computer Science, University of Southampton, United Kingdom
[b] Faculty of Information Technology, Kazakh British Technical University, Almaty, Kazakhstan

HIGHLIGHTS

- A powerful calculus which models high level languages for interacting with Linked Data.
- A concise and novel operational semantics, with a sound and complete notion of operational equivalence.
- A rich algebra over processes that is sound with respect to the notion of operational equivalence.
- Insight into the connection between soundness and completeness of bisimulation and proof theory.

ARTICLE INFO

Article history:
Received 1 February 2012
Received in revised form 16 April 2013
Accepted 3 July 2013
Available online 24 July 2013

Keywords:
Operational semantics
Bisimulation
Linked Data

ABSTRACT

The aim of this work is to verify an algebra for high level languages for reading and writing Linked Data. Linked Data is raw data published on the Web and interlinked using a collection of standards. The main innovation is simply to use dereferenceable URIs as global identifiers in data, rather than a key local to a dataset. This introduces significant challenges for managing data that is pulled from distributed sources over the Web. An algebra is an essential contribution to this application domain, for rewriting programs that read and write Linked Data.

To verify the algebra, a syntax, operational semantics and proof technique are introduced. The syntax provides an abstract representation for a high level language that concisely captures queries and updates over Linked Data. The behaviour of the language is defined using a concise operational semantics. The natural notion of behavioural equivalence, contextual equivalence, is shown to coincide with the bisimulation proof technique. Bisimulation is used to verify that the algebra preserves the operational semantics, hence rewrites of programs using the algebra do not change their operational meaning. A novel combination of techniques is used to establish the correctness of the proof technique itself.

1. Introduction

This work focuses on high level languages for reading and writing data published on the Web [8]. There is a powerful emerging movement to published raw data on the Web and to interlink the published data by using URIs embedded in the data. Data published in this way is referred to as Linked Data [7,12]. The Linked Data movement is gaining considerable momentum as major organisations including the UK and US governments [59] and the BBC [32], adopt associated technologies and principles for publishing valuable data.

The "link" in Linked Data refers to the URI and its usage. The URI is a standardised and globally recognised identifier for resources. Instead of publishing documents, as is done for traditional Web sites, raw data is published directly by

* Corresponding author at: Faculty of Information Technology, Kazakh British Technical University, Almaty, Kazakhstan.
E-mail addresses: ross.horne@gmail.com (R. Horne), vs@ecs.soton.ac.uk (V. Sassone).

0167-6423/$ – see front matter
http://dx.doi.org/10.1016/j.scico.2013.07.005

organisations. URIs are used to identify resources in the published data, thus distinct datasets may unambiguously refer to the same resource. Using URIs in this way is a small conceptual shift. However, this shift enables new opportunities. Since organisations use a single global naming system for identifiers in data, datasets can be interlinked between any organisations. The result is a global dataset, without boundaries or central control, that is constantly changing.

The traditional setting for data is well understood. Without a global naming system such as the URI, each dataset uses its own naming system or key and each dataset is disjoint. Such a system is a closed system, since the boundaries of the data are known; thus classical logic can be used, to determine whether some data does not appear in a dataset for instance. Also, database schemata can be employed to structure the data. Denotational semantics is suited to this setting, where the entire space of the dataset is clearly defined [53].

The Linked Data setting presents significant new challenges. The use of URIs as a global naming system links distributed data sets by allowing them to refer to common resources. Furthermore, a RESTful (HTTP based) protocol [22] can be used to obtain new data from distributed sources based on URIs that appear in known data [21,39]. An HTTP GET request on a URI can be used to obtain data about the resource the URI represents. URIs that return data in this way are called dereferenceable URIs [33]. In this Linked Data setting, there is no guarantee that dereferencing a URI will find all relevant data about a resource. There may always be relevant data published on the Web that is not obtained by the protocol. Thus, in general, maximal query results cannot be expected, nor can classical logic be employed. Furthermore, due to decentralisation, schemata that constrain data cannot be enforced globally. Operational semantics is suited to this setting, where the full context is not known and operations may not be maximal. Also, operational semantics are suited to specifying how Linked Data evolves, as required to model read–write Linked Data [8].

Several technologies for Linked Data standardised by the World Wide Web Consortium (W3C) have found widespread use. These technologies include a light data format called the Resource Description Format (RDF) [40], and the SPARQL query language [30]. RDF is a more loosely structured data format than XML, consisting of RDF triples. An RDF triple resembles the universal idea of a simple sentence in natural language, where a property is used like a verb in natural language to relate a subject and an object. Due to this flexibility, diverse data from different sources can be lifted to RDF. Thus RDF is suited to combining data from many datasets, so that queries can be asked across multiple datasets. Some smart techniques have been employed to implement distributed SPARQL queries [60,61,56].

The authors provided the first operational semantics for RDF and SPARQL Query in their FOCLASA workshop paper [37]. The calculus in this work adapts the calculus in the workshop paper so that it also captures SPARQL Update, which became a W3C recommendation in 2013 [25]. SPARQL Update specifies updates over RDF that delete data, insert data and constrain updates according to queries. The first operational semantics for SPARQL Update was also provided by the authors [38]. This work presents a high level language which internalises RDF, SPARQL Query and SPARQL Update with some constructs for concurrency. The update language reuses constructs from the query language, thus it makes sense to consider this more general language that combines RDF, SPARQL Query and SPARQL Update. In this sense, a new high level domain specific language is proposed. Note that the HTTP operations for harvesting Linked Data are not modelled in this calculus, but they are modelled in related work [21,39]. However, URIs that appear in the electronic version of this paper are real dereferenceable URIs, e.g. *res:Linked_data*.

In this work a natural notion of operational equivalence is used to assign semantics to the terms of a process calculus for Linked Data. An algebra for the calculus is axiomatised using common structures including semirings [28], such that the algebra is sound with respect to the operational equivalences introduced. The equations derived from the algebra can be used to rewrite processes via equational reasoning, which enables quick implementations, essential optimisations and novel programming techniques. Optimisations include finding normal forms for the distribution of updates across collections of datasets, a key problem for Linked Data [31]. Rewriting processes is also relevant to the Big Data application area [1]. For Big Data applications, a process must be rewritten to a form that allows the process to be executed over a fault tolerant, hence scalable, cluster of machines [20].

To be able to express the operational semantics of languages for Linked Data, a novel framework for expressing operational semantics is used. In many ways, the framework goes beyond traditional frameworks for concurrency, such as the π-calculus [50], due to the powerful atomic actions that must be expressed. Indeed the framework raises significant questions about operations for concurrency which are often taken for granted, such as interleaving merge in the presence of synchronisation.

The order of presentation. Sections 2 and 3 explicitly mention the syntactic techniques employed and motivates the applications of an algebra for Linked Data using examples. Readers who prefer to first read the formal definitions can skip straight to Sections 4 and 5. Here the syntax of a high level language and its operational semantics are specified. Section 6 introduces and verifies the proof techniques employed, i.e. the soundness and completeness of bisimulation with respect to contextual equivalence. Finally, Section 7 introduces an algebra over terms in the language and proves the correctness of the algebra with respect to bisimulation.

2. A bridge between operational semantics and Linked Data

This paper spans several areas of computer science. Therefore it is beneficial to explicitly mention the techniques employed. Techniques drawn from the fields of process calculi and logic are applied to high level programming languages